

Simple and Fast Interval Assignment Using Nonlinear and Piecewise Linear Objectives

Scott A. Mitchell
Computing Research
Sandia National Laboratories

International Meshing Roundtable
October 2013

20 minute talk, 5 minute questions



Outline

- **Problem definition**
 - a few simple examples of why it is hard
 - tradeoffs needed
 - may be infeasible
 - an infeasible example
- **Prior approaches**
 - Linear constraints – good
 - Linear objective – not so much IMHO, depending on setting
 - Tam & Armstrong concentrates deviations
 - Lex min max is slow series of problems
- **New solution**
 - Cubic Objective
 - Piecewise linear to get integer solution
 - Implemented

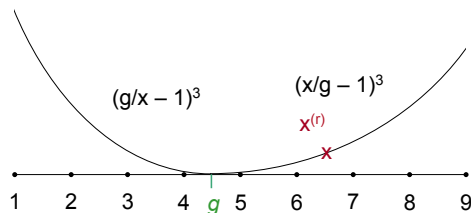
Interval Assignment Summary

Simple and Fast Interval Assignment Using Nonlinear and Piecewise Linear Objectives

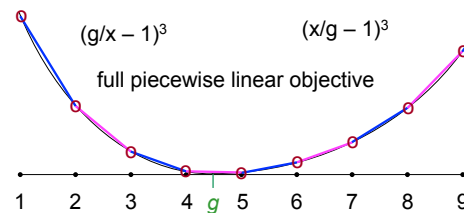
Scott A. Mitchell

- New nonlinear objective function
- Finesse integer constraints using L_1 minimization of convex piecewise linear approximation
 - adapt piecewise, slope
 - heuristics for constraint interactions

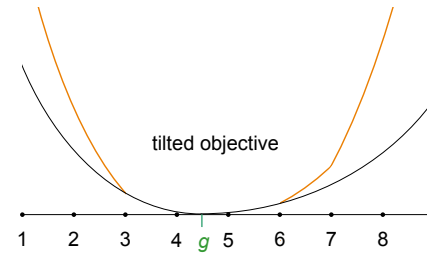
$$\begin{aligned} \min f(x) \\ \text{s.t. } Ax = b \\ x_I \in \mathbb{N} \\ l \leq x \leq u. \end{aligned}$$



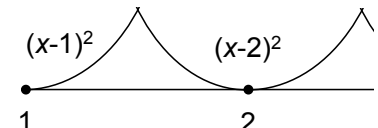
I just relax



bend



tilt



&

wave!



- First known improvement to FEM interval assignment **structure** since 1997 lexicographic min-max
 - Lots of people write new constraints for new schemes, but this is first change to objective (except Graphics community solving related-but-different problem using MIQP)
- Implemented in MeshKit, using IPOPT
 - scales to thousands of surfaces in nuclear reactor core models
 - 1997-2013, Cubit runs lex min-max for every quad and hex mesh
 - New solution may migrate to Cubit, as Cubit includes MeshKit library

Interval Assignment (IA)

Problem Definition

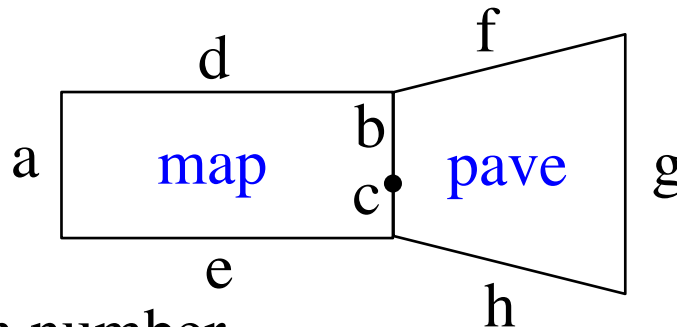
- Set the **number of mesh edges** on each curve, such that the owning surfaces and volumes can be meshed according to their schemes.
 - compatible interface meshes, e.g. for parallel
 - quad and hex meshing
 - triangle and tet meshes have trivial constraints

$$a=b+c$$

$$d=e$$

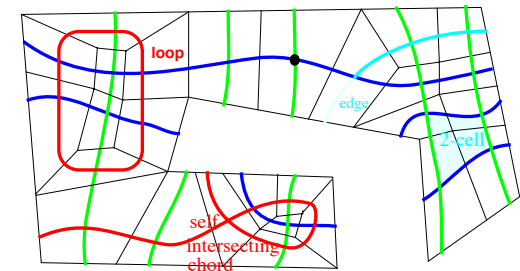
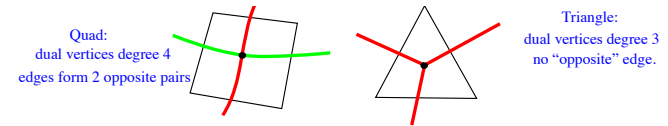
$$b+c+f+g+h=\text{even number}$$

all variables integer (e.g. $a=4$, $b=2.5$, $c=1.5$ is not valid)



Why is Interval Assignment Hard?

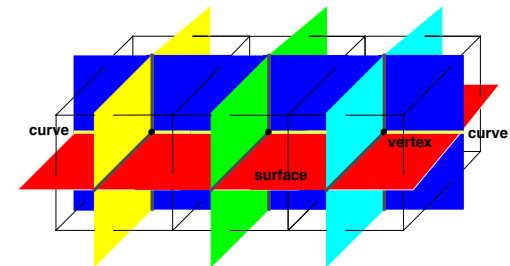
- What's the problem?
 - hex meshing is a global problem
 - quad meshing is a global problem
 - for the same reasons,
 - but with much weaker constraints
- Integrality makes it much harder.
 - Discrete problem
 - Combinatorial algorithms needed



A quadrilateral mesh and the corresponding STC.

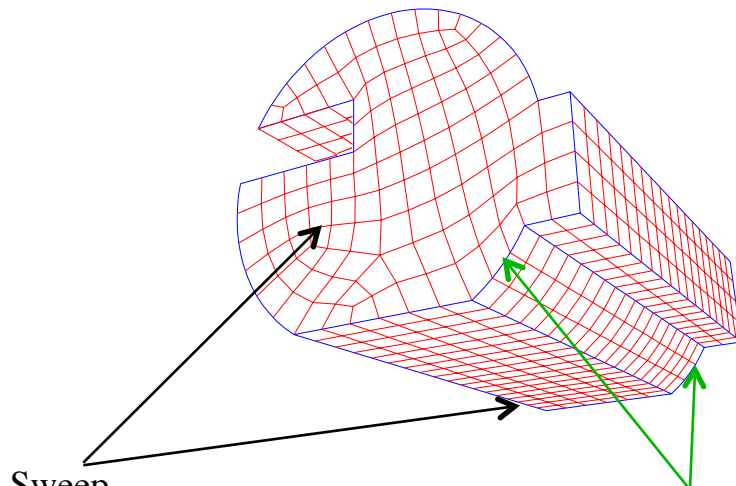
Unstructured quad meshing:

Each chord enters the quad mesh an even number of times (0 or 2) so we need an even number of mesh edges on the boundary. (necessary and topologically sufficient)



Unstructured hex meshing:

Each chord enters the hex mesh an even number of times (0 or 2) so we need an even number of quads on the boundary. Plus contractible loops must be even. (necessary and topologically sufficient)



Sweep
same quad mesh
connectivity front and back

Map
same curve mesh connectivity
top and bottom. e.g. 3 edges



Interval Assignment by linear constraints is old

Interval assignment problem form

- Linear constraints, linear objective
 - Tam & Armstrong 1993
- Linear constraints, lexicographic min-max objective
 - Mitchell 1997
- Fluid flow matching
 - Muller-Hanneman 1997
- Linear constraints, quadratic objective
 - Bommes et al. 2009+

Interval assignment constraints

- Volumes with holes
 - Shepherd et al.
- Midpoint subdivision
- Paver
- Submapping
- Skew control

Automatic Scheme selection

- White & Tautges 2000

Popular to define constraints for new meshing schemes.

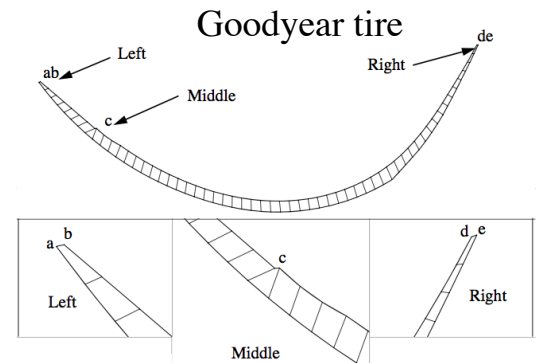
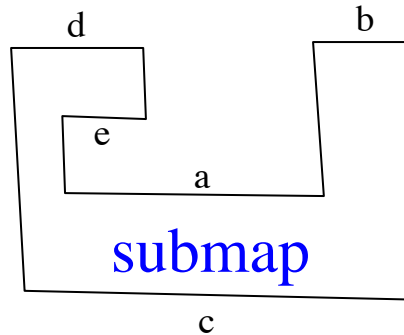
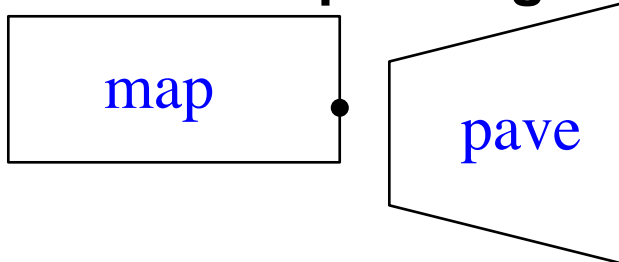
Little change to
objective function form
in FEM community
1997 — this paper

(But active area in Graphics 2009+)

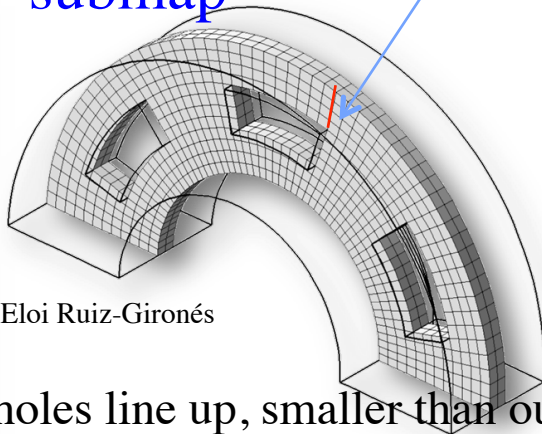
Local Constraints

curve has at least one edge
 $x \geq 1$

- Each surface has local constraints
 - depending on scheme (structure)
 - depending on user desires (sizing, skew)



submap
 Edge length at least 0.1:
 $X \leq 4$



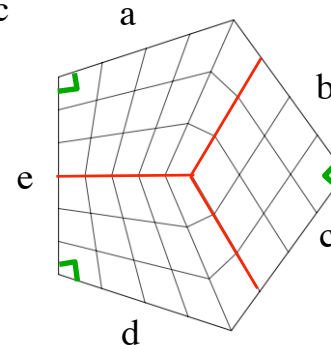
Eloi Ruiz-Gironés

$$d - e + a + b = c$$

$$e \leq d + 1$$

$$a < c$$

$$e < a \text{ OR } \dots$$



midpoint subdivision
 $\text{side1} = a + b$
 $\text{side2} = c + d$
 $\text{side3} = e$
 $\text{side1} \leq \text{side2} + \text{side3}$
 $\text{side2} \leq \text{side1} + \text{side3}$
 $\text{side3} \leq \text{side2} + \text{side1}$



Mixed-Integer Linear Constraints

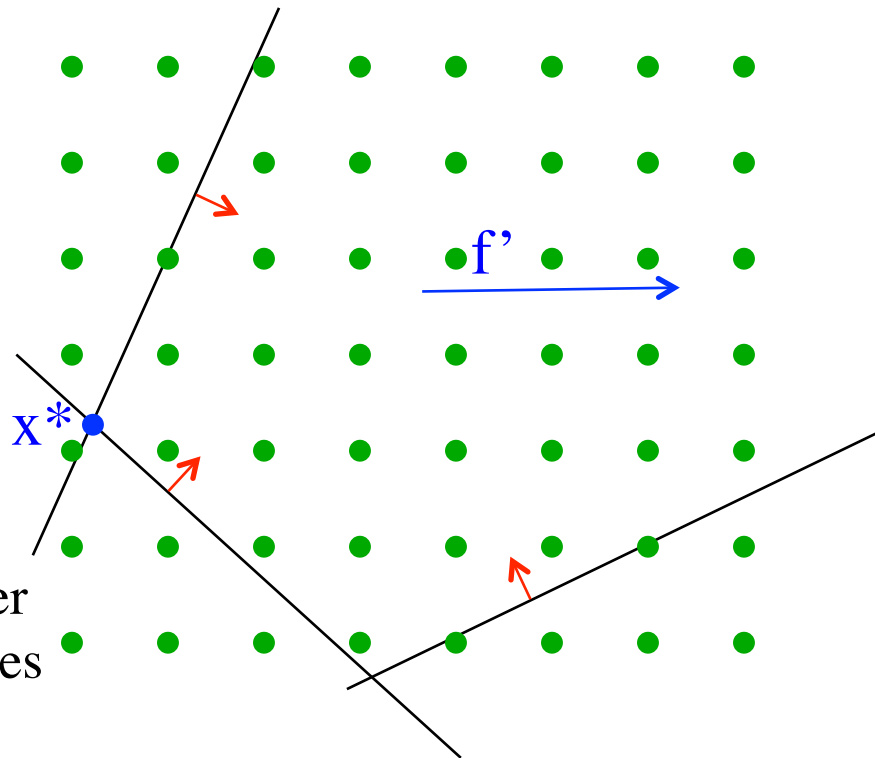
- Constraints are linear, no high powers such as $x^2 = y^3$
- Coefficients are typically 1.
- Sum-even modeled with artificial variable, 2-coefficient
 - $\text{sum}(x_i) = \text{even} \Leftrightarrow \text{sum}(x_i) = 2z : z \text{ is integer}$
- Expressed in matrix form

$$\begin{aligned} &\min f(x) \\ &\text{s.t. } Ax = b \\ &\quad x_I \in \mathbb{N} \\ &\quad l \leq x \leq u. \end{aligned}$$

- Inequality and equality are equivalent
 - $Ax \geq b \Leftrightarrow Ax + y = b, y \geq 0$
 - $Ax = b \Leftrightarrow Ax \leq b \text{ and } Ax \geq b$

Benefits of Linear Constraints

- Feasible region (space of solutions) is convex (possibly unbounded) we have efficient solution algorithms (especially for linear objective)
 - Not so easy for integer solutions

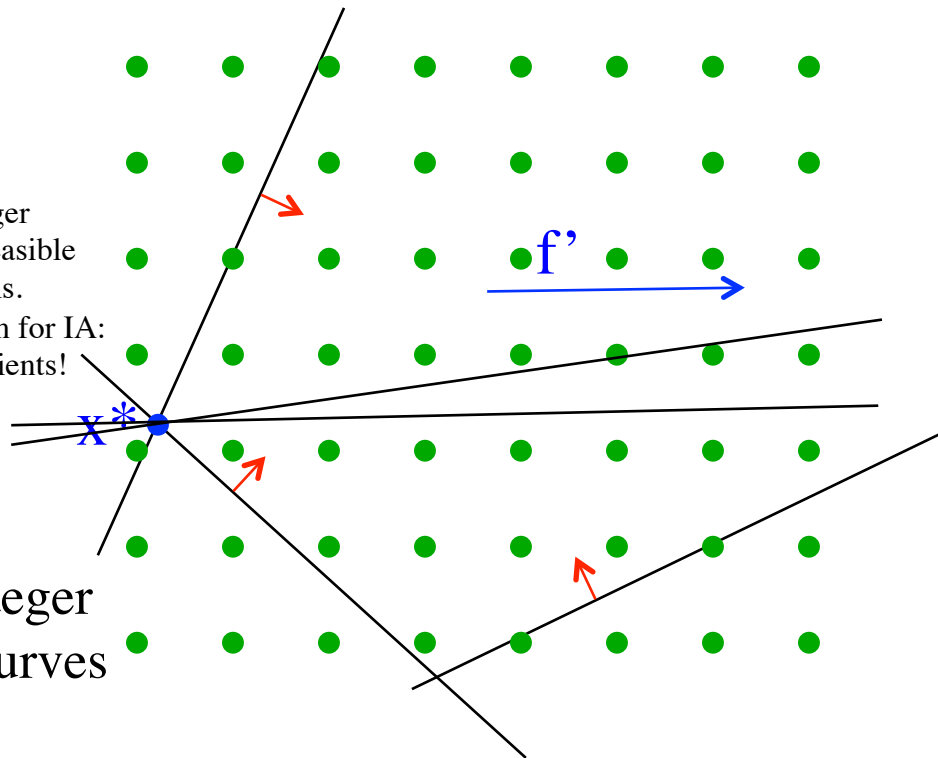


2^v nearby integer
points. $v = \# \text{curves}$

$$\begin{aligned} \min f(x) \\ \text{s.t. } Ax = b \\ x_I \in \mathbb{N} \\ l \leq x \leq u. \end{aligned}$$

Benefits of Linear Constraints

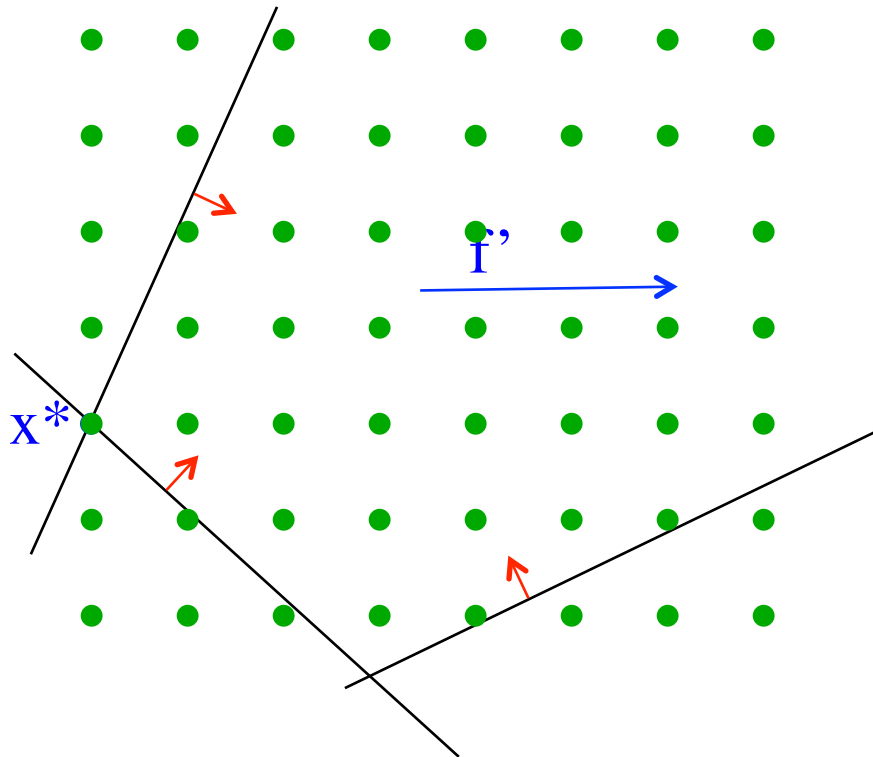
- Feasible region (space of solutions) is convex (possibly unbounded) we have efficient solution algorithms (especially for linear objective)
 - Not so easy for integer solutions



$$\begin{aligned} \min f(x) \\ \text{s.t. } Ax = b \\ x_I \in \mathbb{N} \\ l \leq x \leq u. \end{aligned}$$

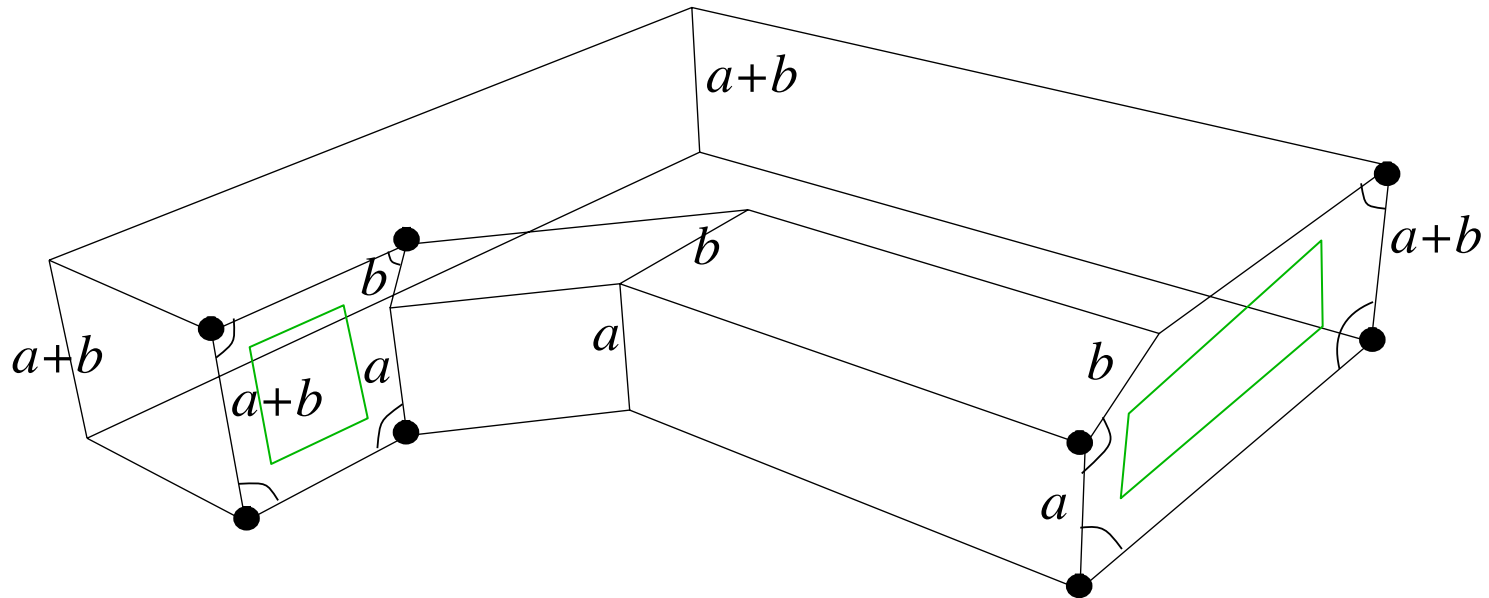
Benefits of Linear Constraints

- Feasible region (space of solutions) is convex (possibly unbounded) we have efficient solution algorithms (especially for linear objective)
 - Not so easy for integer solutions
 - Unless coefficients are cooperative!



$$\begin{aligned} \min f(x) \\ \text{s.t. } Ax = b \\ x_I \in \mathbb{N} \\ l \leq x \leq u. \end{aligned}$$

Global Infeasibility



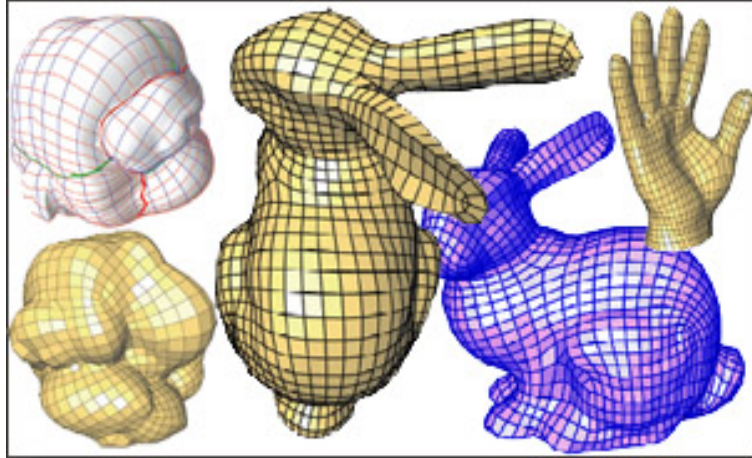
$$a = a + b, b = 0, \times$$

The global IA problem may be infeasible, meaning no solution exists, even when each surface can be meshed in isolation.

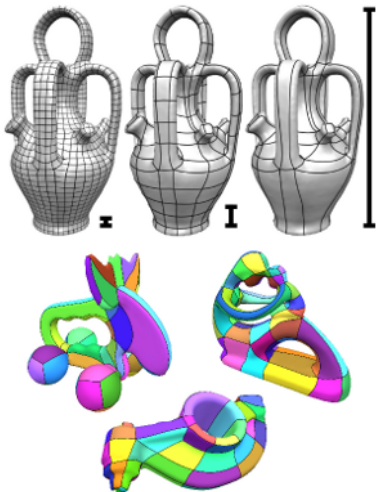
State of the Art in Quad Meshing

Eurographics STARS 2012

Graphics has discovered quad meshes



Designing Quadrangulations with Discrete Harmonic Forms
Yiying Tong, Pierre Alliez, David Cohen-Steiner, and Mathieu Desbrun
ACM/EG Symposium on Geometry Processing 2006, pp. 201-210



Mostly structured quads
on smooth curved surfaces
Key issue is placement
of irregular nodes

Integer-Grid Maps for Reliable Quad Meshing
David Bommes, Marcel Campen, Hans-Christian
Ebke, Pierre Alliez, Leif Kobbelt
SIGGRAPH 2013

Motorcycle Graphs: Canonical Quad Mesh Partitioning

Eurographics Symposium on Geometry Processing 2008

Pierre Alliez and Szymon Rusinkiewicz

(Guest Editors)

http://www.disneyanimation.com/library/motorcycle_sgp_2008.pdf

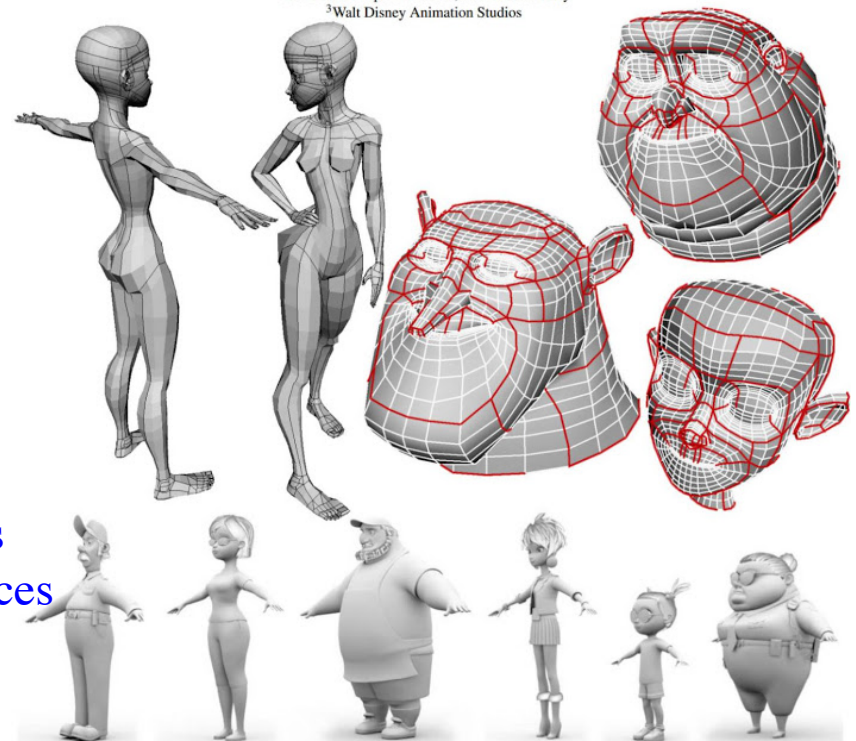
Volume 27 (2008), Number 5

David Eppstein^{†1}, Michael T. Goodrich^{†1}, Ethan Kim², and Rasmus Tamstorf³

^{†1}Computer Science Department, University of California, Irvine

²School of Computer Science, McGill University

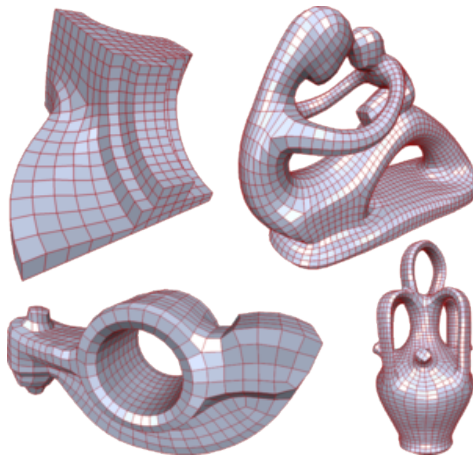
³Walt Disney Animation Studios



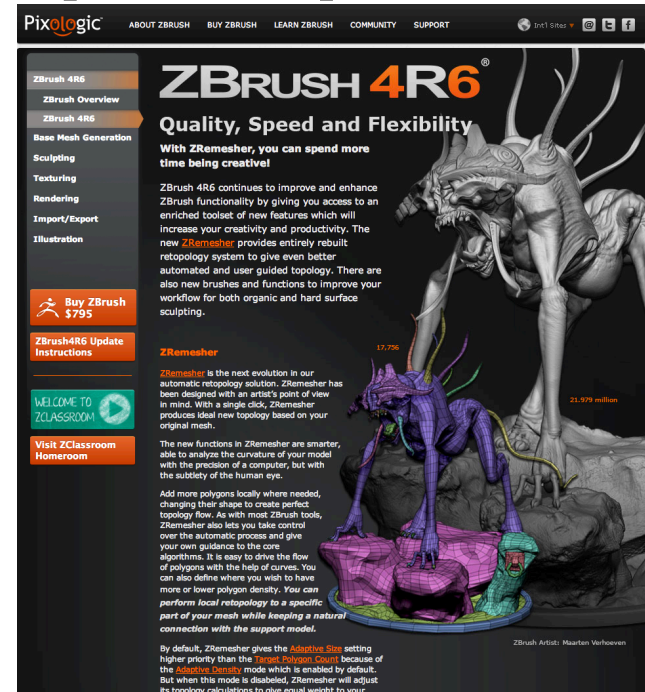
Disney Animation
(Pixar uses triangles)

Graphics has discovered quad meshes including the need for interval assignment

Solved with mixed-integer linear-constraint optimization, series of problems
The placement of irregular nodes is included in the optimization problems



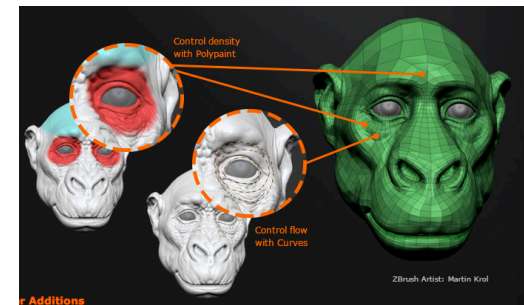
commercial
version



Mixed-Integer Quadrangulation.

David Bommes, Henrik Zimmer, and Leif Kobbelt.

ACM Trans. Graph. (TOG), 28(3):77:1–77:10, July 2009.



Objectives: user specified sizes

- Goal g for each interval

- about g

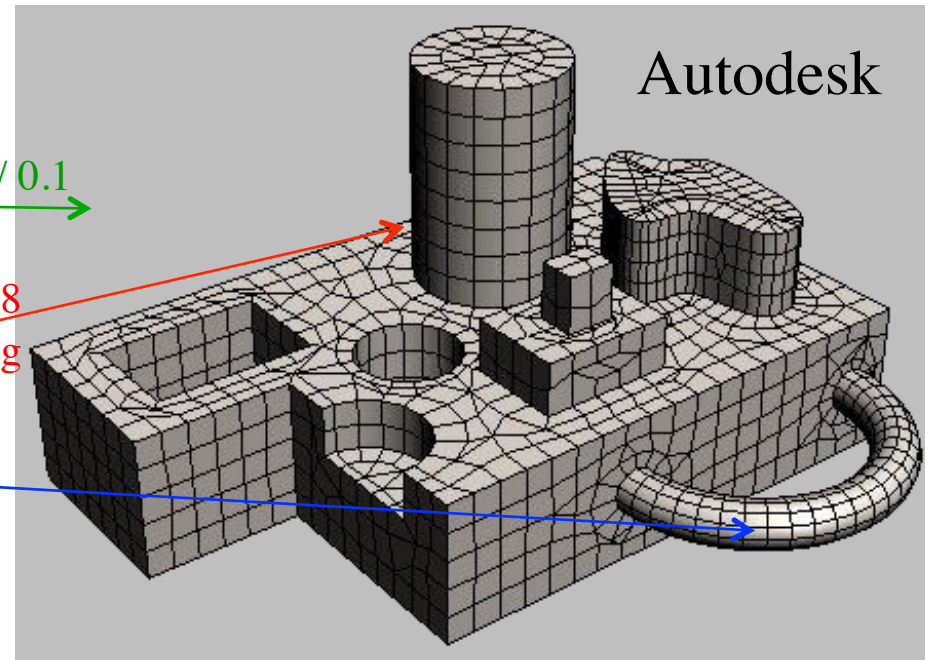
- each mesh edge should be about 0.1 inches long $\xrightarrow{g = \text{length} / 0.1}$
 $x \approx g$

- exactly g (not implemented currently)

- I want exactly 8 intervals here $\xrightarrow{g = 8}$
 $x = g$

- at least g

- I need at least 6 here $\xrightarrow{g = 6}$
 $x \geq g$

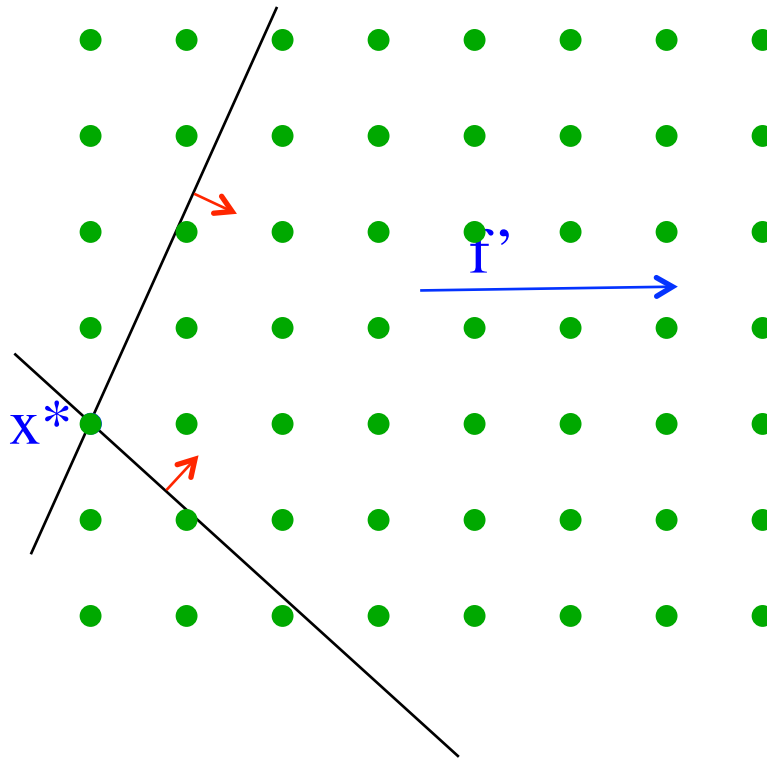


- Achieving all of these g , and constraints, may not be possible.

- How do you measure deviations? Relative values?

What's good about linear objectives?

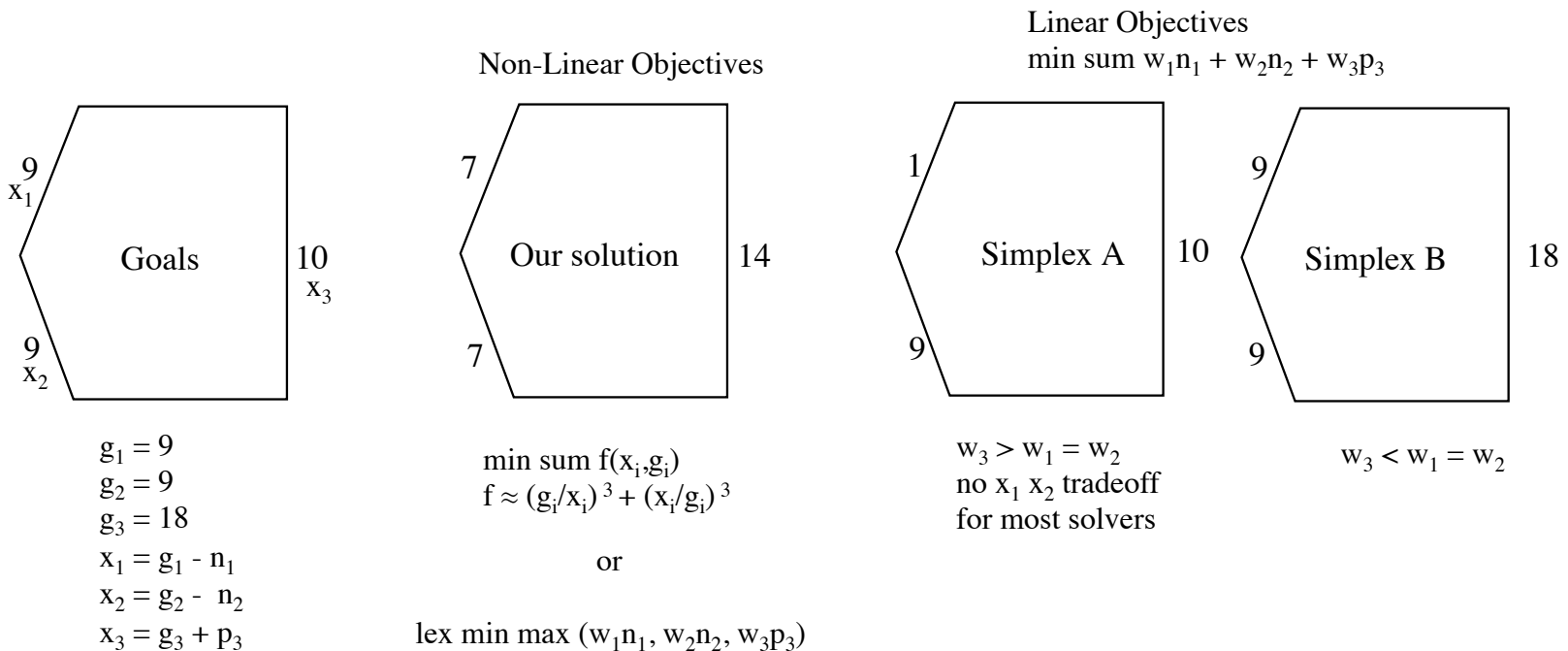
- Optimal solutions occur at corners, easy to find.
 - if constraint coefficients cooperate, corners are integer solutions. Integer solution for free!



$$\begin{aligned} \min f(x) \\ \text{s.t. } Ax = b \\ x_I \in \mathbb{N} \\ l \leq x \leq u. \end{aligned}$$

What's wrong with linear objectives?

- All the deviations are concentrated
 - L_1 minimization leads to sparsity in the solution
 - not a big deal if deviations are small
 - could be a big deal if deviations are large



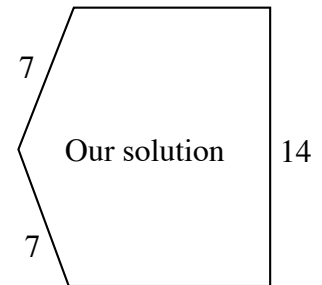


My Solutions

to spread out deviations

- **1997 Lexicographic min max**
solve series of linear programs, one per variable = slow.
 - Minimize maximum $x_i - g_i$. Fix that x_i at a nearby integer value, check feasibility, remove it from equations.
 - minimize maximum remaining $x_i - g_i$, repeat
 - **Weighted deviations, relative size**
 - $\min \max p_i(x_i - g_i > 0) + n_i(g_i - x_i > 0)$,
where $p_i = 1/g_i$ $n_i = 1.3/g_i$
 - i.e. assigning $x=10$ for $g=5$, is as bad as $x=20$ for $g=10$
- **Today, cubic objective (L_3)**
 - A small series of nearby problems to get integer
 - About the same solution as lex min max, depending on problem size
 - L_∞ vs. L_3 norm

lex min max ($w_1 n_1, w_2 n_2, w_3 p_3$)



$$\min \sum f(x_i, g_i)$$
$$f \approx (g_i/x_i)^3 + (x_i/g_i)^3$$

Cubic Objective

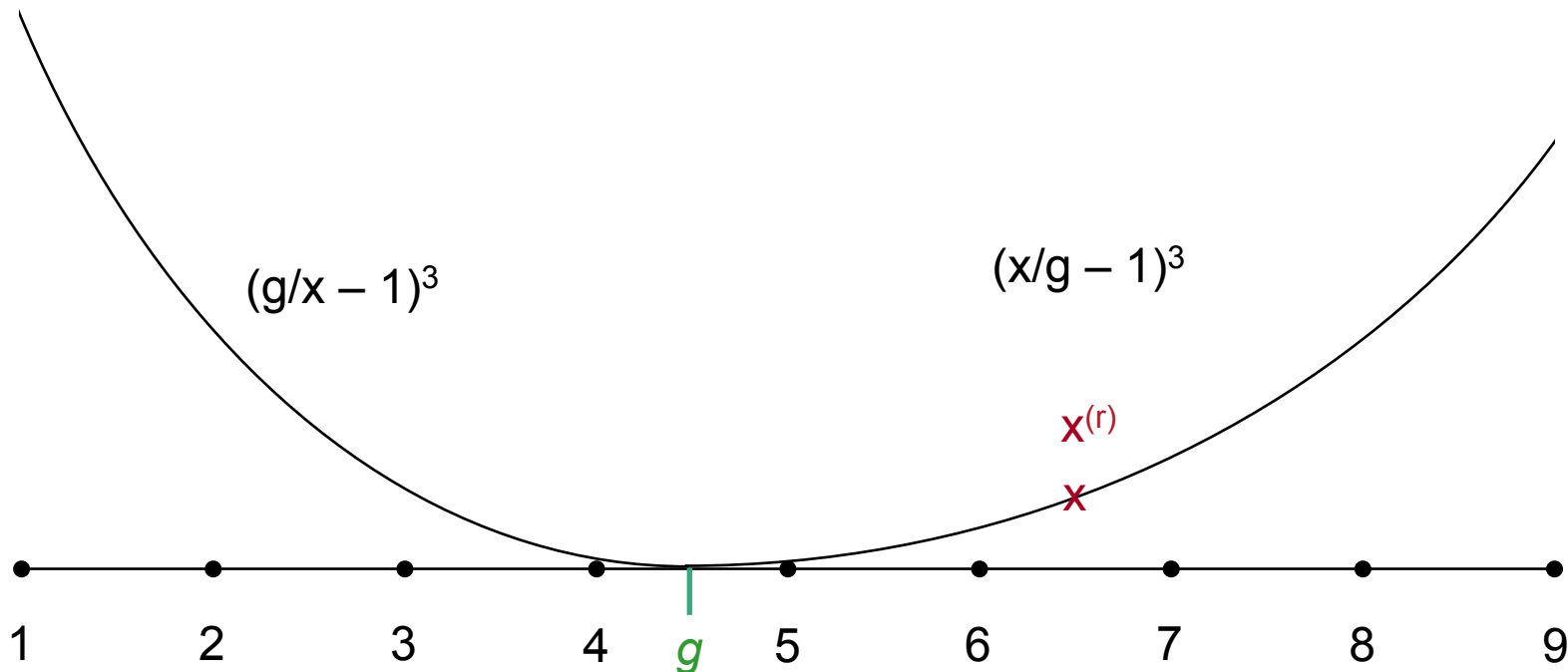
Good: directly measures relative change, ($g \rightarrow x$: 4 \rightarrow 8 is as bad as 4 \rightarrow 2)

Good: no lexicographic min max, single optimization solve spreads deviations

Good: convex objective (plus convex constraints)

local optimum is the global optimum

Bad: non-linear objective requires slower algorithms than a linear one,
but the difference is less today than in 1997



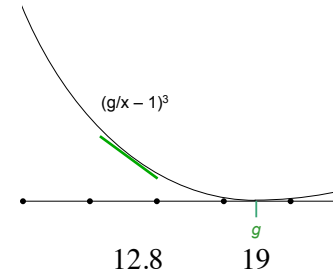
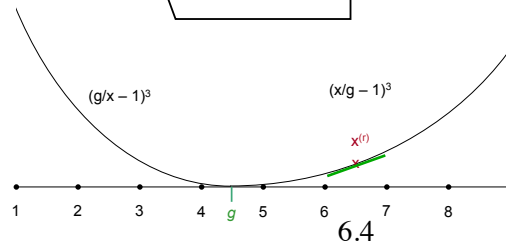
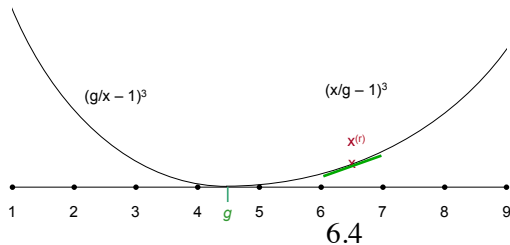
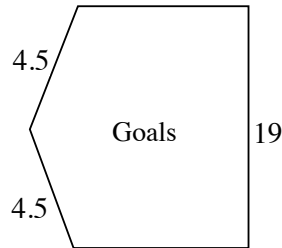
Cubic Objective Solution

Non-integer solution usually

(later problem definitions fix)

$$\min \sum f(x_i, g_i)$$

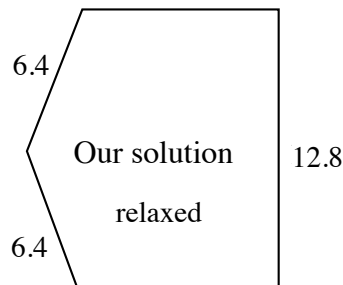
$$f = (g_i/x_i - 1)^3 \text{ or } (x_i/g_i - 1)^3$$



$$\min F = (x_1/4.5 - 1)^3 + (x_1/4.5 - 1)^3 + (19/x_3 - 1)^3$$

minimum when $F' = 0$ (or extremes $x_i = 1$, etc.)

sum of slopes = 0

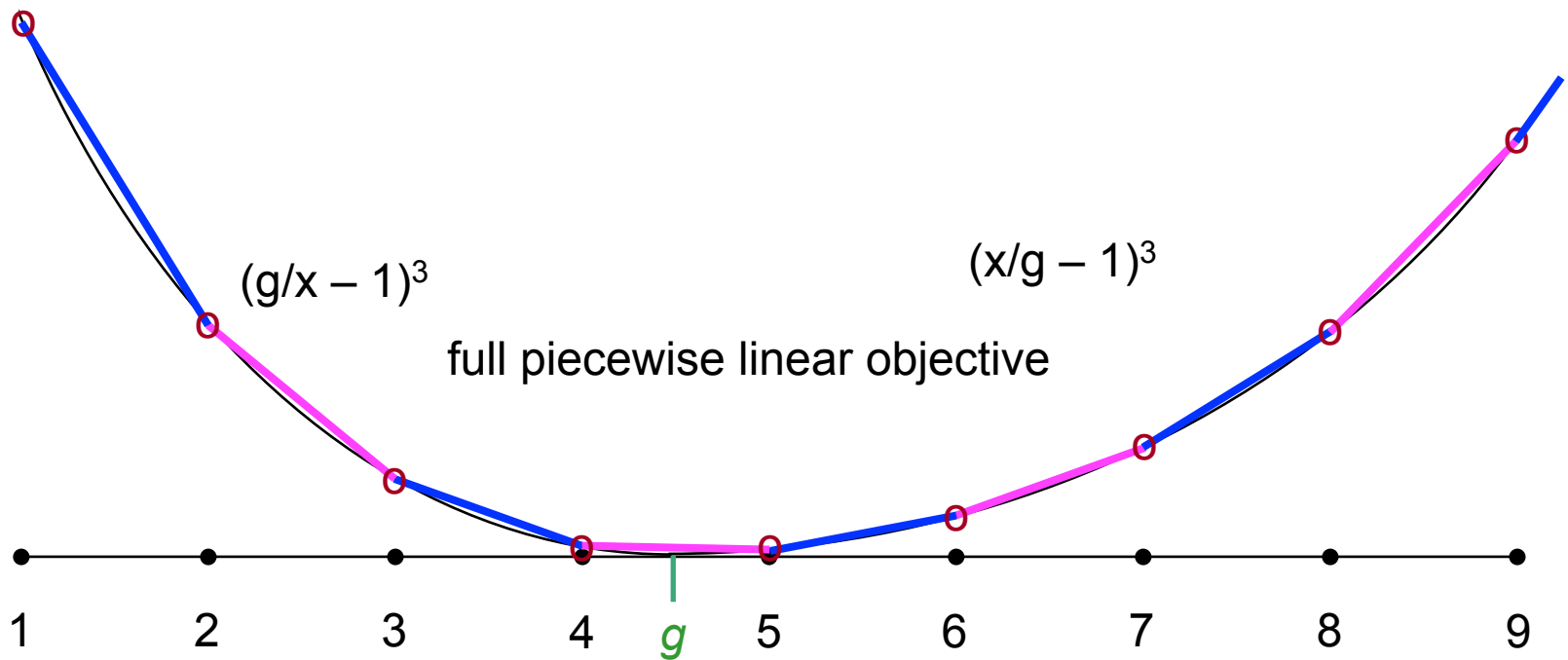


I have spread out the deviations, made the major compromises

Piecewise Linear Objective

Good: restores integer solutions for free (often)

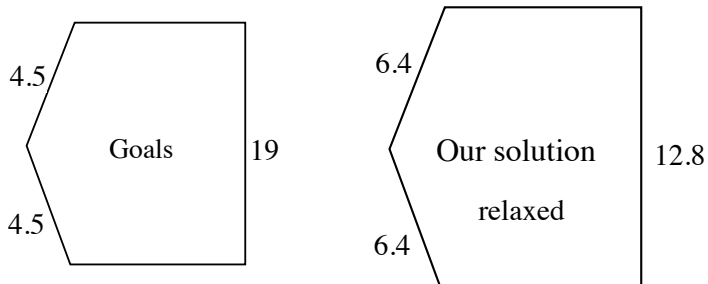
Bad: explicit variable for every single unit interval = expensive time, memory



Bend at integer points \leftrightarrow corner at every integer
in the higher-dimensional linear problem
where each unit interval is a dimension.

Adaptive Piecewise Linear Objective

For efficiency, just add the bends (pieces) as needed



Delta variable for each interval $[k, k+1]$

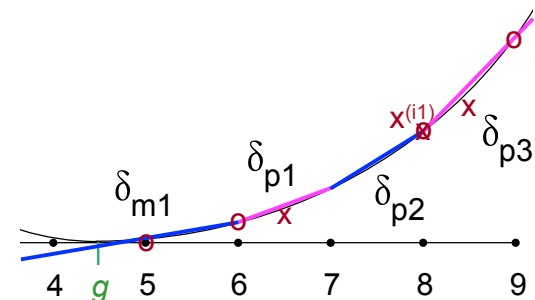
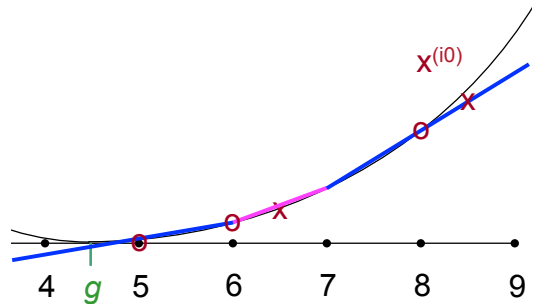
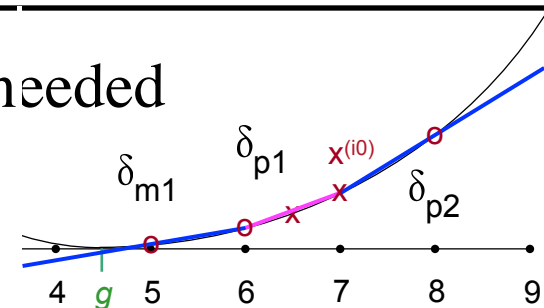
linear slope = weight = $f(k+1) - f(k)$

Create three intervals:

around relaxed x^* , beyond it, below it

Solve min sum weighted deltas (linear)

Add more deltas if solution beyond $k+1$, repeat



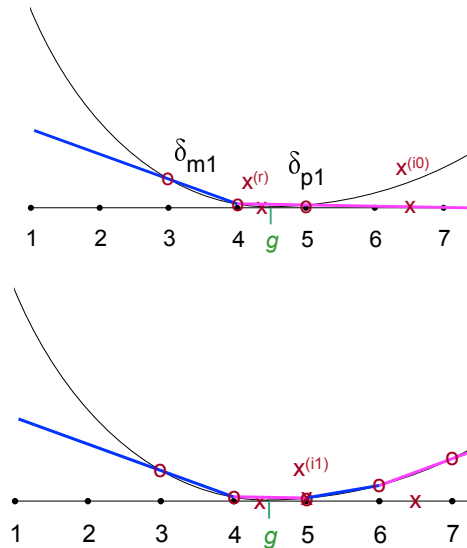
*Notional. For the problem on the left the solution is 6,6,12.

A long chain of surfaces from the long curve might lead to the behavior on the left column.

Adaptive Piecewise Linear Objective

Add the bends (pieces) as needed

Start with two bends, so relaxed solution is in a trough to avoid unbounded solutions right away



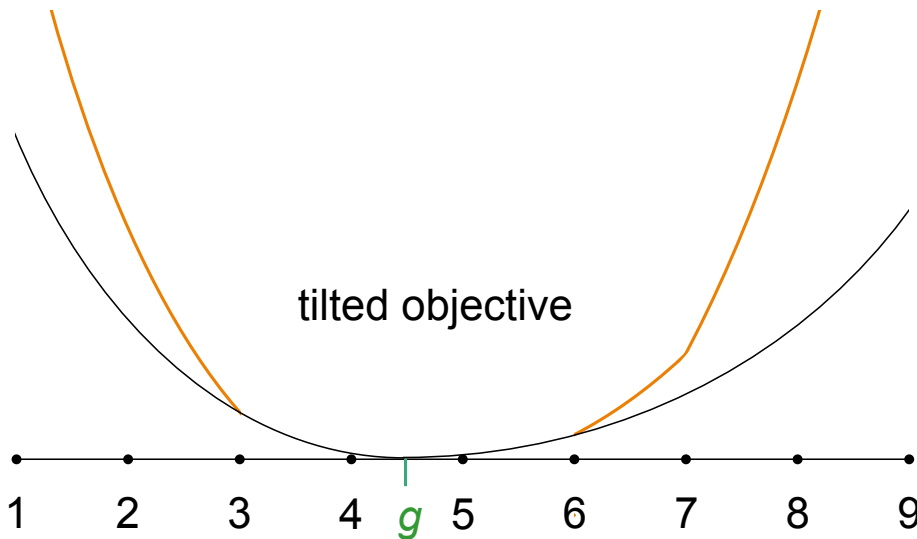
Tilt to Break Non-integer Ties

Break non-integer ties by tweaking the weights (of the deltas), an additive factor to the slope of the objective.

Randomization helps avoid cases where, say, many variables to the right gang up to motivate a non-integer solution on the left.

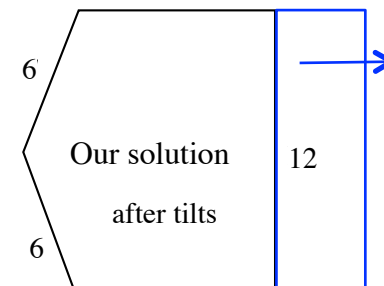
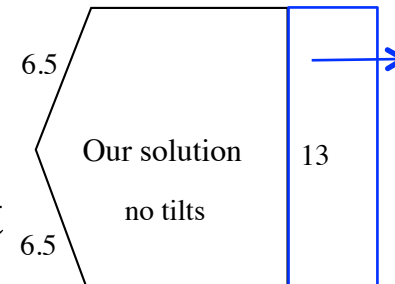
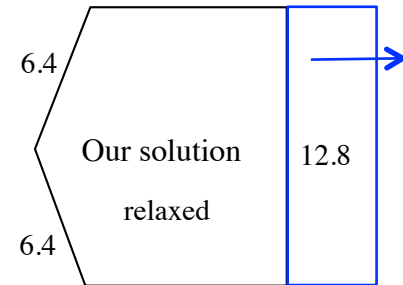
Scaling heuristics needed when goals or deviations vary widely, else small goal variables are left at non-integer values.

Numerical tolerance issue with solvers



e.g.

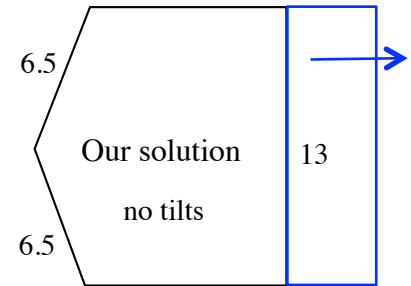
=
constraint



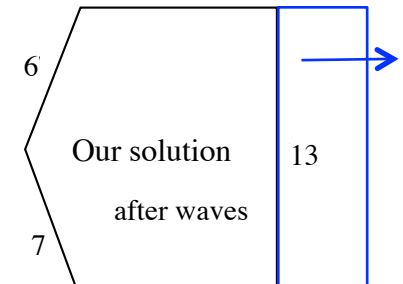
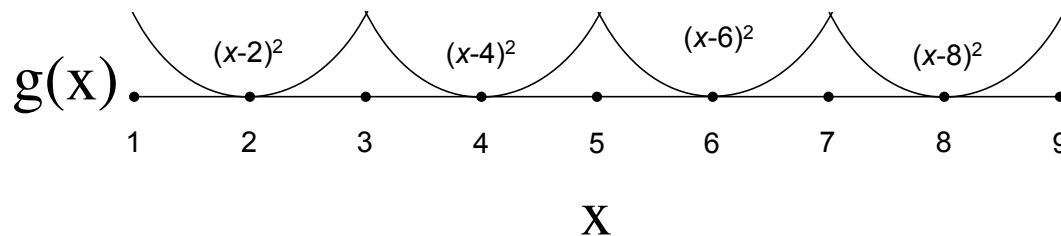
Last Resort, Waves to Force Integrality

unstuck $\frac{1}{2}$ (fractional) integer values

e.g.



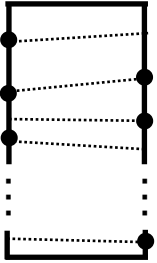
$x = \text{sum of edges}$, need an even number
constrain $g < 0.001$



Bad: breaks convexity, local minima are not global minima
IPOPT has limited global optimization capabilities

Better, in progress: bends and tilts for sum-even variables

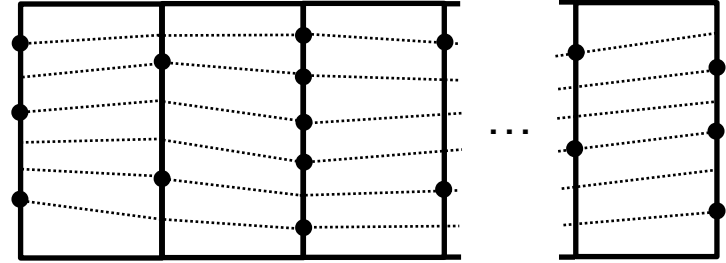
Scaling stress tests



scaling by curves

curves	$x^{(r)}$ s	$x^{(r)}$ c/s	$x^{(i)}$ s	$x^{(i)}$ c/s	$x^{(i)}/x^{(r)}$ time	total s	total c/s
2,000	0.027	74k	0.084	24k	3	0.11	18k
6,324	0.051	120k	0.37	17k	7	0.43	15k
20,000	0.14	140k	1.9	11k	14	2.0	10k
63,238	0.48	130k	7.0	9k	15	7.5	8k
200,000	2.0	100k	50	4k	25	52	4k
632,378	11	57k	430	1.5k	39	444	1.4k

Runtime about $O(c^{1.5})$ for c variables.



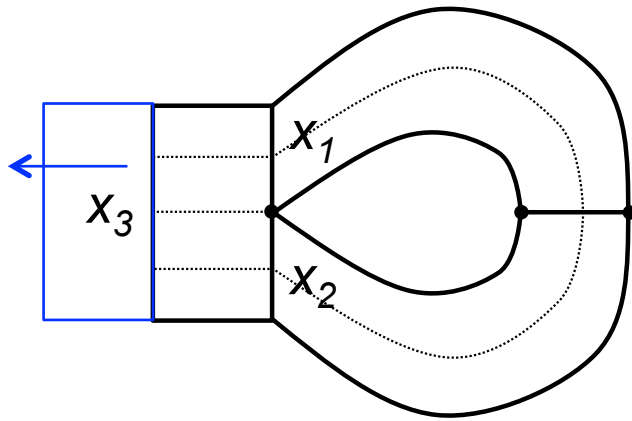
scaling by faces

faces	curves	$x^{(r)}$ s	$x^{(r)}$ f/s	$x^{(i)}$ s	$x^{(i)}$ f/s	$x^{(i)}/x^{(r)}$	total s	total f/s
160	1,061	0.028	5700	0.061	2600	2.2	0.091	1800
505	3,298	0.067	7500	0.21	2400	3.1	0.27	1900
1,600	10,614	0.23	7000	0.74	2200	3.2	0.98	1600
5,050	32,793	0.80	6300	2.3	2200	2.9	3.2	1600

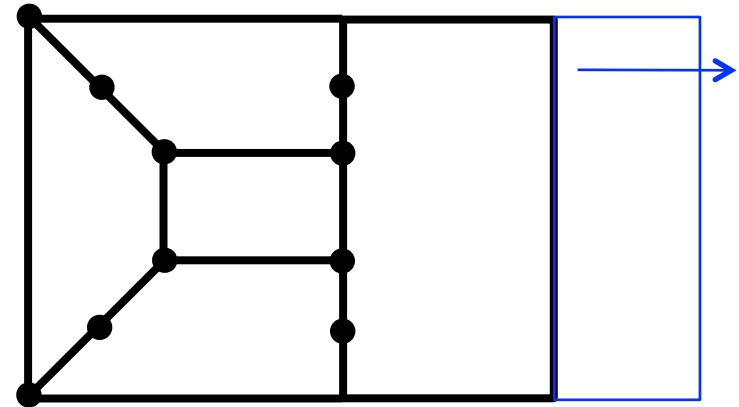
Time for an *integer* solution is only a small multiple of the time for the *relaxed* solution.
In contrast, Lex min max grows quadratic+ in problem size

Robustness stress tests

Problems needing tilts to get integer solutions



“radish”



Split curves provide more degrees of freedom:
more tilts are needed to remove fractional values.

Only a few tilts are necessary,
tilt for each split curve in the chain is simultaneous.

Availability and Status

- **MeshKit algorithm implementation**
 - **Open source**
 - <http://gnep.mcs.anl.gov:8010/meshkit/>
 - **Well defined API**
 - **IPOPT optimization library is free**
 - **MA27 linear algebra library is free**
 - **but you have to download it yourself**
 - HSL (formerly the Harwell Subroutine Library). A collection of Fortran codes for large scale scientific computation. <http://www.hsl.rl.ac.uk>, 2011. Includes MA27. IPOPT special instructions at <http://www.hsl.rl.ac.uk/ipopt/>
- **To do:**
 - **Robustness and quality testing on large models**
 - **Tilt for sum-even constraints**
 - better than just waves

[Main Page](#) [Related Pages](#) [Namespaces](#) [Classes](#) [Files](#)

%MeshKit: A Library for Mesh Generation

0.9


MeshKit is a library for geometry-based 2d and 3d mesh generation. This library has two main purposes:

- to provide a useful collection of open-source mesh generation functions
- to provide infrastructure (geometry, lower-dimensional entity meshing, etc.) for implementing new meshing functionality

MeshKit uses a component-based approach, using external components for selected functionality where possible. Geometry and relations to mesh are accessed through the iGeom and iRel interfaces, resp. Mesh is accessed through both the MOAB and iMesh interfaces. MeshKit allows registration of external meshing algorithms, allowing those algorithms to operate in collaboration with the rest of MeshKit functionality.

This manual is divided into the following sections:

- User's Guide
 - [Using MeshKit: The 2-Minute Introduction](#)
 - [Data Model](#)
 - [A Graph-Based Model for Mesh Generation](#)
 - [A more detailed example of graph traversal](#)
 - [More Graph-Based Meshing Examples](#)
 - [Geometry, Mesh, Relations Interfaces](#)
 - [MeshKit API](#)
 - [MeshKit Algorithms](#)
- Developer's Guide
 - [Coding Style Guide](#)
 - [Geometry, Mesh, Relations Interfaces](#)
 - [Writing A New MeshOp Class](#)
 - [Frequently Asked Questions](#)

Generated on Thu Mar 3 15:45:39 2011 for MeshKit by  1.5.5



Alternatives

- **Why bother with the nonlinear solver?**
 - it is fast, uses little memory
 - but maybe you don't have one
- **Skip the relaxed cubic-objective phase**
 - start with piecewise linear adaptive bends
 - efficiency would depend on numerics, how far the optimal solution is to the initial goals



Alternatives

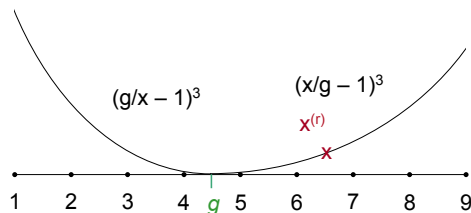
- **Why sum-of-cubes objective function?**
 - Any power in $[2, \text{infinity}]$ is possible.
 - Lex Min Max from 1997 is a form of L_{infinity}
 - The lower the power, the bigger the worst deviation, but the fewer number of curves need to change
 - Experimental. I tried powers of 2, 3, and 4.
 - 2 concentrated deviations too much
 - 4 spread deviations out too much in some contrived cases
 - This was just my opinion
 - Some other power may be better, depending on what a “typical” model is for you.

Interval Assignment Summary

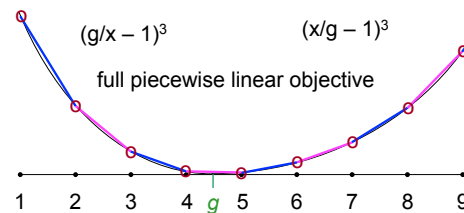
Simple and Fast Interval Assignment Using Nonlinear and Piecewise Linear Objectives

Scott A. Mitchell

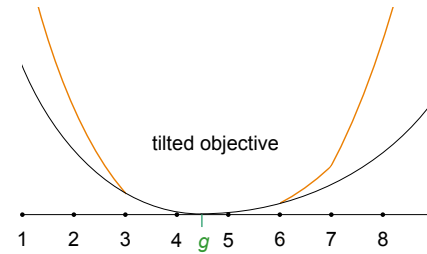
- New nonlinear objective function
 - Finesse integer constraints using L_1 minimization of convex piecewise linear approximation
 - adapt piecewise, slope
 - heuristics for constraint interactions
- $$\begin{array}{ll} \min f(x) \\ \text{s.t. } Ax = b \\ x_I \in \mathbb{N} \\ l \leq x \leq u. \end{array}$$



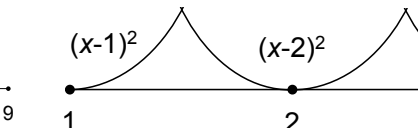
I just relax



bend



tilt



&

wave! 😊

- First known improvement to FEM interval assignment **structure** since 1997 lexicographic min-max
 - Lots of people write new constraints for new schemes, but this is first change to objective (except Graphics community solving related-but-different problem using MIQP)
- Implemented in MeshKit, using IPOPT
 - scales to thousands of surfaces in nuclear reactor core models
 - 1997-2013, Cubit runs lex min-max for every quad and hex mesh (that meshes the boundary first)
 - New solution may migrate to Cubit, as Cubit includes MeshKit library



Thanks

It takes a village to make me successful

- Thanks to Tim Tautges for suggesting and supporting this project
- Thanks to Tim Tautges, Rajeev Jain, for MeshKit
- Thanks to Carl Laird, for IPOPT
- Thanks to David Bommers for Graphics quads via mixed-integer quadratic programming
- This work was funded under the auspices of the Nuclear Energy Advanced Modeling and Simulation (**NEAMS**) program of the U.S. Department of Energy Office of Nuclear Energy, U.S. Department of Energy. Partial support for this work was provided through Scientific Discovery through Advanced Computing (**SciDAC**) program funded by U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.
- Is the government still shut down?
Will mesh for food.